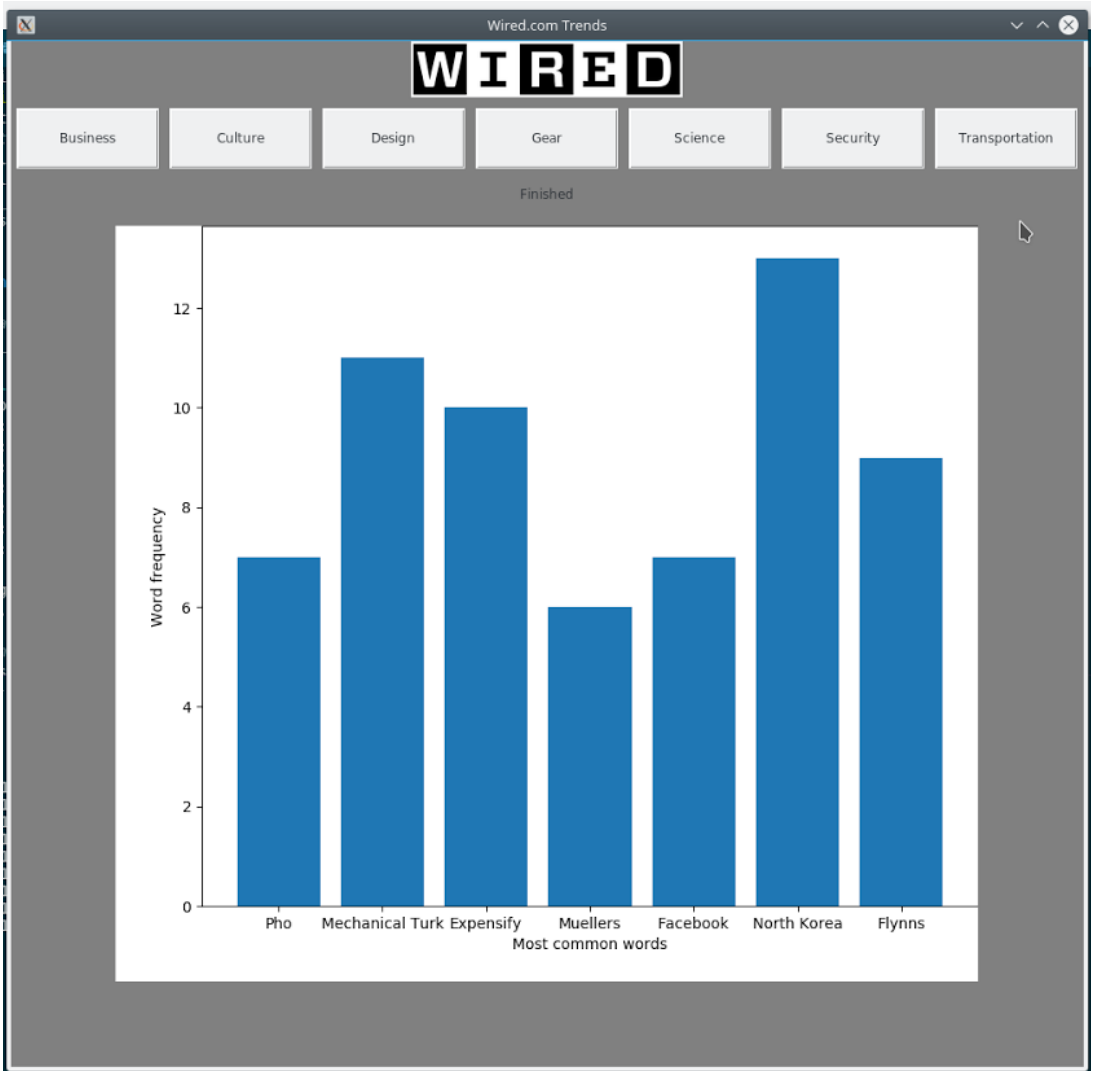
Christopher Jantzen
William Osbourne
Christopher Morrison

Introduction:

Our application is a python program used to get the word consistency trends from any section on wired.com. We wanted a straightforward interface, which includes the Wired log, and seven buttons, one for each section of the website. After you press the button, the program uses the input to determine which url to use. Each article url is then scraped from the page, and each url is sent to a function which takes the text and stores it in a file. The file is used for data analysis to find the word that appear the most through all the articles in the section. The text is analysed against a list of 20,000 common english words such as “this”, “the”, and other similar words. The words are then totalled and displayed in a bar graph to show the top seven words used in the section.



Project Results:

We found the results to be favorable, as we accomplished everything that we had set out to do. Naturally, it wasn't without some trials to get to that point. For example, in the web scraping stage, Will faced problems when scraping unicode characters such as "thin spaces" and emoji from the website to put into a standard text file.

Chris J. experienced problems finding a way to implement a bar graph that would be able to be changed each time a button was pressed to show the new words and change the graph accordingly. The graph was also slightly too small, causing larger words to overlap with the labels next to them, making it very difficult to read what the words were.

Chris M. was fortunate enough not to encounter any errors in python. However a small difficulty did arise from trying to determine how to filter keywords to find the most important as there were quite a few. In the end the most used proper nouns were determined to be the most important as they typically top the word count and are the subject matter of the articles.

As for future improvements we could add, we discussed being able to change the graph types to either bar charts, pie charts to show the relative comparison of the words. We could also expand the scope of the project to include a breakdown of word frequency by each article and show those results with the total analysis of the section. Additionally, we could establish a baseline for certain words that the wired writers may commonly use and only display words that have appeared more than normal in previous articles. Another potential feature is linking the article that mentions the keyword the most back to the wired.com website so that the user can read the articles about the subject.

Looking back, this project was effectively a very simple web scraper combined with a graphical interface to show the user. This could be useful for tracking trends on more websites than just wired.com. We could potentially track the price of certain goods at kroger over the year and know when the best time to buy certain foods is. Another way we could perhaps use it is to look for certain phrases or trends of phrases to identify if a news cycle is politically leaning one way or another.

Division of Group Work:

In a general breakdown of the work, we divided the program into three main sections. Interface, which was handled by Chris Jantzen. Data Analysis, handled by Chris Morrison. Web Scraping, handled by Will Osbourne.

Chris Jantzen -

I created the user interface using python's Tkinter GUI package. Within this interface, I used the PIL library to add the logo of Wired.com and the "partial" function from the functools library for function calls with each button. I also used functionality from the matplotlib and pylab packages for implementing a bar graph that shows the top seven most commonly used words from each section of the website. I believe that my interface is functional and serves the purpose it needs to, while still looking aesthetically pleasing and well spaced. Figuring out how to implement the plot using adjustable data required a good amount of research and trial and error. I watched a series of Youtube tutorials on the basics of Tkinter and how to implement things like buttons, canvases, and padding, as well as some videos on matplotlib to learn the mechanics of using MATLAB in python. I also read through the official documentation of the packages, which allowed for me to better shape and style the window and each of the components. The interface shows the Wired.com log with seven buttons below it corresponding to each section of the website. Below the buttons is a message log that changes as specified when analysing the data and sending it to the bar graph. The graph shows the most used words along the x-axis and the number of time the word appeared on the y-axis.

Chris Morrison -

My section of the software was the data filtering tool. The data filtering functions would read in a text file and return a dictionary with the word count of words that followed certain rules. These rules included such things as making sure that a word wasn't one of the 20,000 most common words used in the English language. Additionally all sequential words that were capitalized were read in as one key value, assuming that they were a title. Words that appeared to be urls, numbers, or only appeared three or fewer times were also filtered out of the results. Using the end result we could see the trends in names of people, products, and companies that were making the news.

In addition to the data filtering I was also in charge of setting up the python virtual environment and bash run scripts. These were done so that any user would be able to immediately run the project without having to find all the dependent libraries and install them. Because python is so portable, this also allowed us to share our dependencies a lot faster than if we were to each install them individually. I feel that this is equally as important as the software project itself as we could develop the most useful software in the world but if it only ran on my local machine, it would be worthless.

Will Osbourne -

My section, web scraping, was heavily using the library BeautifulSoup to parse the html from Wired. This was not the only library that I had to use, I also needed to use the request portion of urllib to get the data from the web page itself. I, personally, feel that I did a good job on my portion of the program, as I got it in a semi working state to help my group members be able to understand and integrate my section into their portion of the code. After that was in place, I continued to work to overcome unicode issues I found, as well as properly integrating with my team members code, and leading the creation of the lab document.

Bibliography:

Chris Jantzen -

- Libraries Used:
  - https://docs.python.org/3/library/tk.html
  - http://www.pythonware.com/products/pil/
  - https://docs.python.org/3/library/functools.html
  - https://matplotlib.org/api/pyplot_api.html
  - https://matplotlib.org/faq/usage_faq.html
- https://stackoverflow.com/questions/2177504/individually-labeled-bars-for-bar-graphs-in-matplotlib-python
- https://pythonprogramming.net/how-to-embed-matplotlib-graph-tkinter-gui/
- https://www.youtube.com/watch?v=RJB1Ek2Ko_Y&list=PL6gx4Cwl9DGBwibXFtPtflztSNPGuIB_d (entire 14 video playlist)
- http://jakevdp.github.io/mpl_tutorial/tutorial_pages/tut1.html
- https://www.youtube.com/watch?v=QqQ6AEx7QJI

Chris Morrison -

- https://stackoverflow.com/questions/21107505/word-count-from-a-txt-file-program

Will Osbourne -

- Libraries Used:
  - https://docs.python.org/3/library/urllib.html
  - https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- https://stackoverflow.com/questions/17732695/how-to-return-plain-text-from-beautiful-soup-instead-of-unicode

Code Appendix:

- https://github.com/ChristopherMorrison/Super-Spice